LAMP-TR-005
CFAR-TR-850
CS-TR-3739

January 1997

The Detection of Duplicates in Document Image Databases

David Doermann, Huiping Li, Omid Kia, Kemal Kilic

Language and Media Processing Labratory
Instititue for Advanced Computer Studies
College Park, MD 20742

Abstract

Document imaging technology has developed to the point where it is not uncommon for organizations to scan large numbers of documents into databases with little or no index information. This may be done for archival purposes, in which case the necessary index may be as simple as a case number, or with the ultimate goal of automatically extracting index information for content-based queries. Maintaining the integrity of such a database is difficult, especially in a distributed environment where copies of documents with different physical histories may be scanned at different times. In this paper we present a novel approach to detecting duplicate documents in very large databases using only features extracted from the image. The method is based on a robust "signature" extracted from each document image which is used to index into a table of previously processed documents. The system is able to deal robustly with differences between scanned documents with respect to such factors as resolution, skew and image quality. The approach has a number of advantages over OCR or other recognition-based methods including speed and robustness to imaging distortions. To justify the approach and demonstrate its scalability, we have developed a simulator which allows us to change parameters of the system and examine performance while processing millions of document signatures. A complete system has been implemented and tested on a collection of technical articles and memos.

***The support of the LAMP Technical Report Series and the partial support of this research by the National Science Foundation under grant EIA0130422 and the Department of Defense under contract MDA9049-C6-1250 is gratefully acknowledged.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JAN 1997**	2. REPORT TYPE	3. DATES COVERED **00-01-1997 to 00-01-1997**

4. TITLE AND SUBTITLE **The Detection of Duplicates in Document Image Databases**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**40**	

LAMP-TR-005
CAR-TR-850
CS-TR-3739

MDA9049-6C-1250
February 1997

The Detection of Duplicates in Document Image Databases

David Doermann
Huiping Li
Omid Kia
Kemal Kilic

CAR-TR-850 MDA9049-6C-1250
CS-TR-3739 February 1997

The Detection of Duplicates in Document Image Databases

David Doermann, Huiping Li, Omid Kia and Kemal Kilic

Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
doermann@cfar.umd.edu

Abstract

Document imaging technology has developed to the point where it is not uncommon for organizations to scan large numbers of documents into databases with little or no index information. This may be done for archival purposes, in which case the necessary index may be as simple as a case number, or with the ultimate goal of automatically extracting index information for content-based retrieval. Maintaining the integrity of such a database is difficult, especially in a distributed environment where copies of documents with different physical histories may be scanned at different times.

In this paper we present a novel approach to detecting duplicate documents in very large databases using only features extracted from the image. The method is based on a robust "signature" extracted from each document image which is used to index into a table of previously processed documents. The system is able to deal robustly with differences between scanned documents such as resolution, skew and image quality. The approach has a number of advantages over OCR and other recognition-based methods including speed and robustness to imaging distortions.

To justify the approach and demonstrate its scalability, we have developed a simulator which allows us to change parameters of the system and examine performance while

The support of this effort by the Department of Defense under contract MDA 9049-6C-1250 is gratefully acknowledged.

processing millions of document signatures. A complete system has been implemented and tested on a collection of technical articles and memos.

Keywords: Document Image Databases, Duplicate Detection, Shape Coding, Document Indexing

1 Introduction

Steady increases in computational power and affordable storage have allowed very large heterogeneous databases to be considered as a viable means of archiving or storing document images. It is not uncommon to see document collections in excess of one million images, and in today's distributed environments, the management, storage, and retrieval of these collections have become important issues. One primary consideration is in the generation of the index information used to identify the database object. Traditional database indexes may contain, for example, administrative data, document ID numbers, and possibly a small number of keywords which are provided or can be extracted directly from the data. Image databases, however, typically require the manual entry of index information since adequately expressive indexes are not alway available. When dealing with millions of documents, manual indexing is typically not cost-effective.

A second concern is that in an image-based system, traditional organization, search and retrieval techniques are not ideal, in part because of the sheer volume of information that images contain. Although documents are a written representation of a language, a document in image form lacks type content information typically available with text documents. If accurate and unique index information is available for images, many of the operations which are handled by traditional database systems is trivial. In cases where index information is not available, indexing and retrieval remain difficult and challenging research problems. In this paper we explore one such problem, the problem of detecting duplicate document images, in the absence of appropriate index information.

Consider a situation where thousands of documents are being imaged and added to a single heterogeneous database, possibly from a distributed environment. If multiple instances of the same document exist, they may be re-processed or re-entered unnecessarily. This redundancy in the database may not be desirable for a number of reasons, including increased storage cost, difficulties in maintaining database integrity, increased processing cost for database operations, and cost of indexing multiple images with the same underlying content.

It is therefore desirable to have a preprocessing mechanism for detecting duplicate instances of a document image prior to either indexing it or adding it to a database. Naturally, the definition of a "duplicate instance" is open to some interpretation, and may be defined differently for different applications. The level of *similarity* between documents is a key factor in identifying what constitutes a duplicate. We will define three levels of similarity.

A first level of similarity contains *identical* documents. These are documents which are, for all practical purposes, the same even at the image level. These documents most often result from a scan of a single original manuscript, and multiple copies of the document's image file being distributed electronically. We can assume that the images vary only in the file format or file representation and should be easily identified by performing either a byte-by-byte file or a pixel-by-pixel image comparison. A second level of similarity contains *image-variant* documents which are images that arise from different instances of the same original document. The originals may have been scanned at different times, and may have independently undergone various types and levels of physical degradation. This is often observed, for example, with published documents that were originally distributed to multiple sites (i.e. technical reports, copies of memos, etc.). Image-variant duplicates are identical with respect to content and structure, but may differ substantially at the pixel level and cannot be easily identified from pixel-by-pixel comparisons. A third level of similarity contains documents which vary in structure, but contain essentially the same content. This is common, for example, when a document is originally scanned and entered, and later, a revised or reformatted version of the same document is added to the database. We will not address the problem of *structure-variant documents* since the extent of variation is open-ended.

We are currently addressing the problem of detecting image-variant duplicates, where multiple instances of an effectively identical original source are scanned for incorporation into a database. The original documents may have been written on, stapled, torn, taped, or may have pages missing or a cover added. The document may have been copied repeatedly, so different-generations of copies may be involved. The document may have been scanned at different times and on different devices, so resolution, illumination, and contrast may also be issues. Skew and translation may result in additional distortion.

The goal of our project is to analyze documents which are candidates for being added to a database, so that when variations of an existing document are presented, the system is able to identify the duplicates and not process them further. The ability to process documents without prior indexing is essential if the practical use of large-scale document image databases is to be successful.

In Section 2 of this paper, we provide a brief overview of the problem of duplicate document image detection and our proposed approach. In Section 3, we discuss the systems feasibility and present results obtained using a simulator which allows us to test indexing mechanisms that involve millions of indices. In Section 4 we provide details of our implementation and interface. Finally, we discuss experimental results on a database of technical articles and memos in Section 5 and provide a brief discussion in Section 6.

2 Duplicate Document Identification

2.1 Problem Overview

Let us assume that document images are scanned with the intention of adding the images directly to a database. Depending on what information is available a priori in the system, the problem of duplicate detection can be approached in a number of ways. If, for example, basic index information such as the document number, date, title, authors or number of pages is entered manually, this information could serve as a preliminary filter for duplicates. In most cases, however, high-volume operations prohibit such manual entry prior to scanning. Instead, we would like to identify duplicates directly from their images. Although we have basic quantitative information such as the number of pages, at this point we consider only the analysis of the image itself.

One possible solution which has been proposed is to apply Optical Character Recognition (OCR) to the document image and match as much text as possible between the documents. Although this matching can be done relatively quickly, OCR performance suffers on degraded documents in terms of both accuracy and speed. For this reason, we do not feel that OCR is feasible as a first-level filter, but OCR may be used as a secondary filter, to reduce the number of possible matches from hundreds to tens of documents.

Any approach which we choose should have a number of properties for it to be considered as a feasible solution in this domain. First, we are constrained by the fact that we may be dealing with millions of documents, many of which may be highly degraded. To cope with such situations, we must have a *signature* for each document which is

Robust - The signature should be reliably extractable, even when the document becomes degraded.

Unique - Although we cannot realistically expect the signatures to be unique unless we use an excessively large feature set, a given signature should be associated with several tens of documents at most.

Compact - The storage capacity required to hold the signatures of millions of documents may be very large, so the index keys should be as small as possible.

In addition, the algorithms which extract the signature must be

Fast - Algorithms which take minutes to extract a signature, and then attempt to match it against each document in the database, are not acceptable. The application demands rapid extraction, and near constant time indexing into the database of previously entered documents.

Scalable - Initially the algorithms will work on hundreds of documents, but as more documents are processed, and non-duplicates added, the size of the database could grow to tens of millions.

Accurate - It is acceptable to miss a small percentage of duplicates since the result is simply that the same document is entered twice, but identifying documents as duplicates when they are not (false alarms) is not acceptable.

The overall goal of a duplicate detection system should be to either 1) determine that the document is not a duplicate or 2) accurately identify 10-20 documents of which it could be a duplicate. In the latter case, other methods of analysis such as OCR, structural analysis, or human verification can be used as post-processors to eliminate non-duplicates and confirm duplicates.

Our preliminary experiments have allowed us to draw some general conclusions about the types of algorithms that will and will not work. First, algorithms which attempt any in-depth analysis of the document (such as OCR or structural analysis) are not ideal because they cannot extract the features (recognized characters, in the case of OCR) robustly enough for degraded documents, and because the resulting index information would be too extensive. Second, any matching scheme which requires comparison of the extracted features to a significant portion of the database will fail because of the computational requirements. An efficient indexing scheme is required.

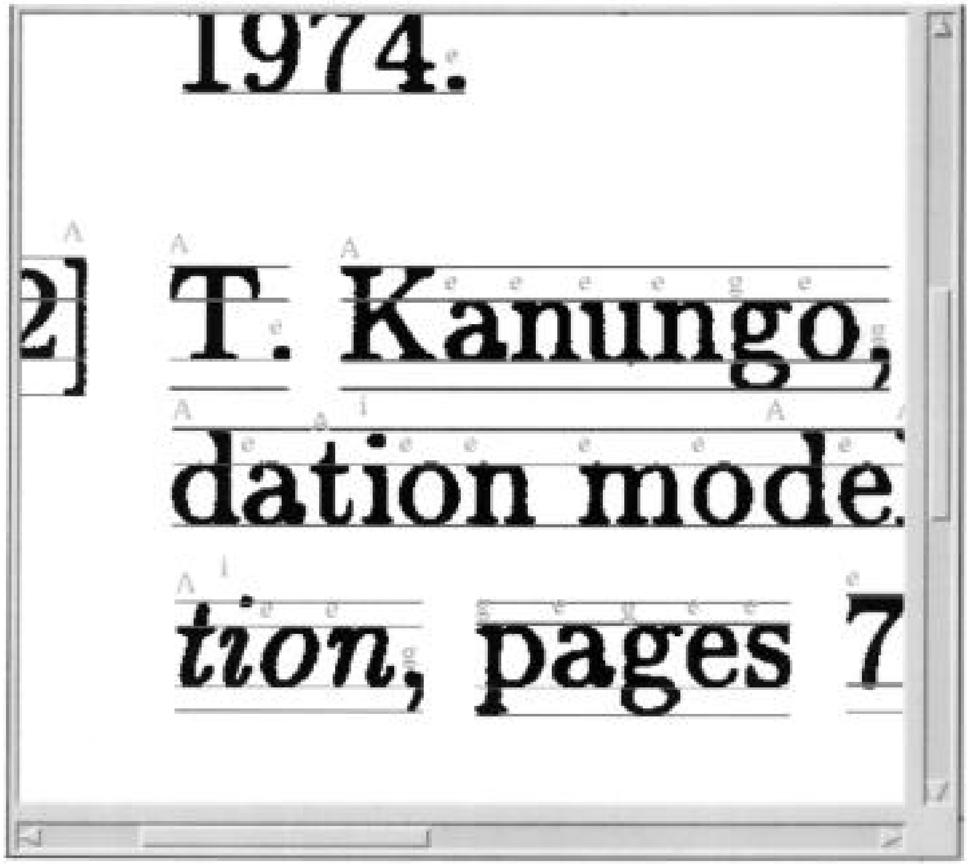
We have developed an approach which promises to fulfill most of our criteria. The approach is based on a robust signature consisting of shape codes extracted from the textual components of the document. In the next section, we describe our approach, some experiments we have run to show its effectiveness, and research issues which must be addressed in order to develop the working system.

2.2 Basic Approach

Our approach is based on the extraction of a signature from a representative line of text in the document image using a shape coding technique. The technique has been used by a number of authors including Tanaka [8] and Spitz [7] for other document analysis applications. Shape coding labels the symbols in the line of text based on very simple shape properties, such as whether the symbols are ascenders, descenders, limited to the x-line, multi-component, or punctuation, for example. These properties are much more robust to noise than the features necessary for OCR, and can be extracted fairly rapidly.

To extract the signature, the document is scanned for a representative sample of text, typically on the order of 50 symbols, on a single line or across several lines. From this sample, the base-line, x-line, ascender-line and descender-line are identified, and each character component is assigned a shape code as shown in Figure 1.

The string of shape codes assigned to the characters in the text sample is used as a signature for the document. A level of robustness is added by indexing based on n-grams of the shape code string, rather than attempting to use an index based on the entire string. Each shape code n-gram serves as an index *key* into the database. A



1974.

2] T. Kanungo,

dation mode

tion, pages 7

Figure 1: Sample character shape code assignment

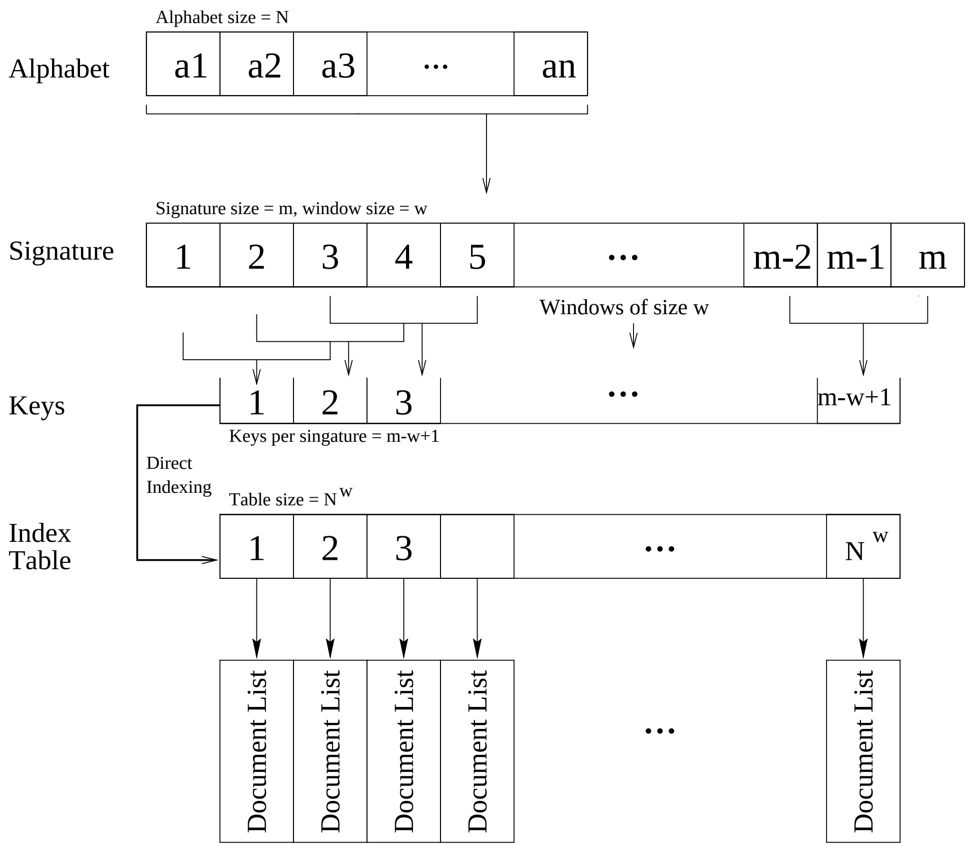


Figure 2: Overview of the indexing scheme.

single dropped or inserted code may affect at most n of these keys but will not affect the entire signature. Figure 2 shows the relationship between the signature, its keys, and the database. When a set of keys is presented for indexing, each key results in a collection of hits from the database. Each hit is a vote for a document, and a ranked list of documents can be returned.

Clearly, a number of additional issues should be addressed in developing a system that satisfies the criteria set forth above. These include:

- use of global classifiers - number of pages, page component statistics, etc. as first-level filters to reduce the duplicate search space.

- choice of a shape code alphabet - selection of features to incorporate into the signature which provide maximum discrimination.

- extraction of features - how to select the signature in the image of a document.

- database organization and indexing - how to create efficient ways to index into large collections.

- verification of candidate duplicates

All of these issues will be addressed in the design phase.

2.3 Related Work

The detection of duplicate or near-duplicate documents has been a problem of interest for some time in many fields, but has not been addressed until recently for collections of images. Some example domains include Education, for detection of plagiarism [5]; Publishing, for detection of unauthorized copies [3, 6]; Databases, for maintaining database integrity; Information Retrieval, for information filtering [9]; and in the USENIX community, for detecting duplicate files [10].

In the document community, most of the work on identifying similar documents has been done using either ASCII documents, or "water-marked" electronic representations. Much less work has been done with document images. A notable exception is Hull [2], who describes a method for matching documents which have the same character content but which may have been reformatted or distorted prior to re-imaging (content-variant documents). Hull's approach represents each document by a set of robust local features which can be used to hash into a database of descriptors. The features in both the query example and the database must be invariant to geometric distortions; by extracting multiple descriptors from each document, they can also be made robust to errors in feature extraction. The measure of similarity is simply the number of features the query document and the database instance have in common. Experiments were performed using as features the character counts for each word in short sequences of words; this provided a set of simple yet robust features that was adequate for small databases. With as few as ten features, 100% accuracy was obtained for a clean query string and a small clean database. Unfortunately, as the number of documents grows, the character count metric becomes less discriminatory.

3 System Feasability

3.1 Theoretical Analysis

Before implementing and testing our approach on real data, we performed a theoretical analysis to see if our design is realistic and if it is robust to errors in signature extraction. The parameters that were varied in our analysis included system-dependent variables such as file size limitations of the operating system, disk access time and disk transfer rate; database variables such as the number of documents, the size of the index table and the average size of the documents; and algorithm variables such as the size of the signature alphabet, the size of the signature and the key or n-gram size.

The analysis yielded qualitative estimates of the expected size of the database, the computational requirements for matching signatures, and the number of missed and false duplicate detections as functions of the database size. It was found that the system could be implemented with generally available hardware.

The details of the analysis are given in Appendix A.

3.2 Simulation Analysis

To demonstrate the technical feasibility of our approach to coding indexing and retrieval we performed several experiments using ideal and corrupted shape code data. A simulator was developed which allowed us to explore a variety of coding, indexing and database organization scenarios, without the need to address feature extraction issues. The goal was to show that signatures can be obtained which are unique enough to be used for indexing and that the database of indexes scales appropriately.

Our simulator takes as input *ideal* ASCII text and maps the characters deterministically into appropriate shape codes, thus simulating perfect feature extraction (see Figure 3). Using the resulting signature, we can explore various indexing options, and test the uniqueness of the signatures on large databases. Since text databases are widely available, a large-scale system can be simulated at relatively low cost.

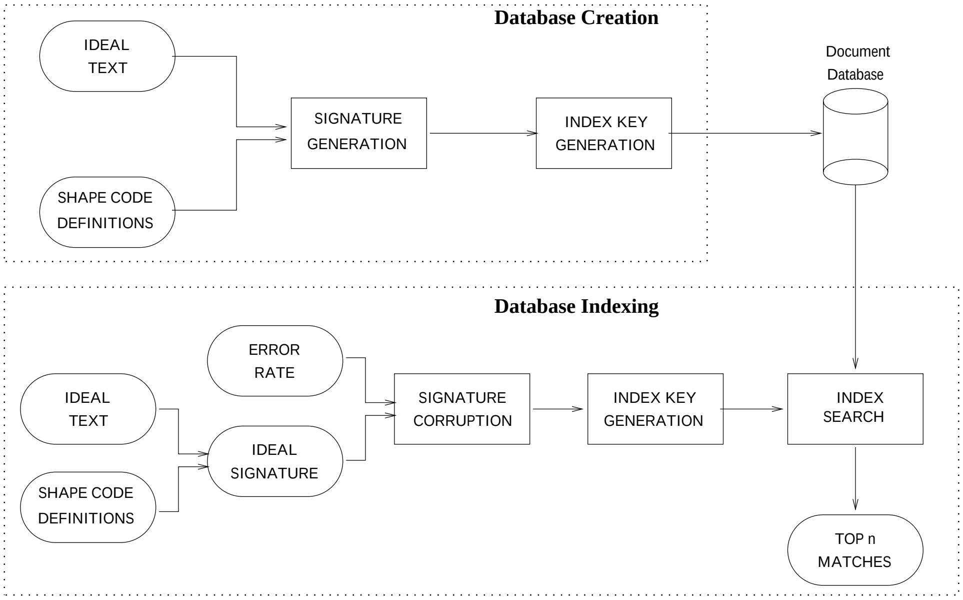


Figure 3: Simulator overview

3.2.1 Experiment 1

In our first experiment, we used a small database so that we could examine and track individual signatures through the system. We extracted 5000 lines of text from an electronic version of the Wall Street Journal each of which contained at least 50 symbols. Each line was treated as an independent document so the database would represent 5000 document images (i.e. 5000 signatures of length > 50). We chose a shape code alphabet of size 8 (shown in Table 1), and a key length of 5 (i.e., a set of 5-grams was generated from each signature). An example of a text line is shown in Figure 4 along with its shape code signature and some of the defined index keys. The keys were then stored in the database.

Our indexing experiments were divided into two parts. Part I was designed to examine the distribution of scores for known duplicates, corrupted duplicates and non-duplicates matched against the database. This would give us an idea of the uniqueness of the signature keys. Part II was designed to look at the distribution of the ranks of retrieved duplicates in the top 20 positions. This would allow us to judge the effect of noise on individual instances. To address the robustness of feature extraction, we further enhanced the simulator by building a noise and degradation model into the system.

Table 1: Table of shape codes and symbols to which they apply

Index	Shape Code	Description	Members
0	-	space	blank
1	X	ascender	! % ( ) 1 2 3 5 7 < > ? C E F G H I J K L M N S T U V W X Y Z [ ] f h k l t
2	y	descender	, . _ y
3	e	xline	* + - : ; = c m n r s u v w x z
4	A	ascender with hole	# $ & 0 4 6 8 9 A B D O P Q R b d
5	g	descender with hole	g p q
6	a	xline with hole	a e o
7	i	ascender mark	i j ” ’ ˆ ‘

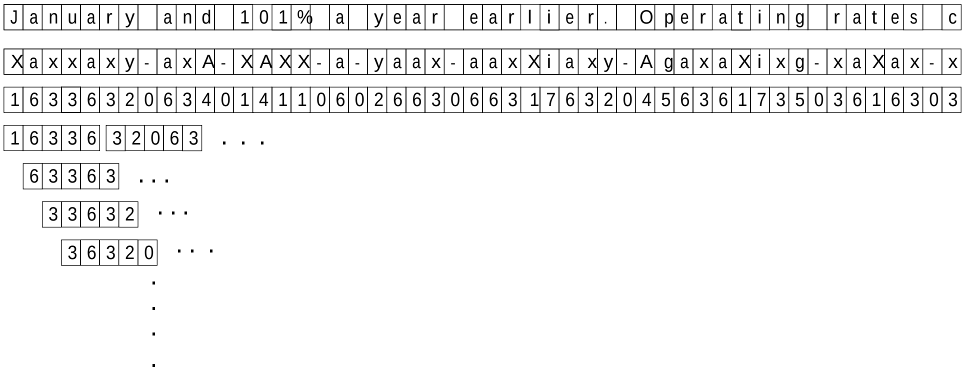

J a n u a r y a n d 1 0 1 % a y e a r e a r l i e r . O p e r a t i n g r a t e s c
X a x x a x y - a x A - X A X X - a - y a a x - a a x X i a x y - A g a x a X i x g - x a X a x - x
1 6 3 3 6 3 2 0 6 3 4 0 1 4 1 1 0 6 0 2 6 6 3 0 6 6 3 1 7 6 3 2 0 4 5 6 3 6 1 7 3 5 0 3 6 1 6 3 0 3
1 6 3 3 6 3 2 0 6 3 . . .
6 3 3 6 3 . . .
3 3 6 3 2 . . .
3 6 3 2 0 . . .
.
.
.
.

Figure 4: Shape codes from sample signature **831** and its index k

The model corrupts the data by randomly perturbing a fix
each signature, thus simulating errors in shape coding.
the database was clean and we applied the model prior to
documents.

In Part I of the first experiment, a set of signatures kn
first matched against the database. Results of matching a
shown in Table 2 for the cases when 0, 5, 10, 15 and 20 er
see that the highest-ranked match is the correct documen
15 errors (i.e. 30% of the symbols corrupted). Table 2 shows that
score drops considerably when there are only 5 errors, th
significantly higher score than alternative choices. The
scores suggests that the match is not simply random, but
candidate.

To test the robustness of the features, we also matched
that were not in the original database (i.e. documents k

Table 2: Scores of line **831** in [candidate (score)] format.

Rank / *Error*	1	2	3	4	5	6	7	8	9
0	**831**(46)	3984(16)	839(14)	3734(14)	834(14)	3752(13)	828(13)	3990(13)	3749(12)
5	**831**(25)	708(10)	218(10)	1029(10)	2788(14)	2789(13)	4474(10)	4498(10)	834(9)
10	**831**(25)	1990(7)	3827(7)	3984(7)	742(6)	3909(6)	2235(6)	4118(6)	541(5)
15	**831**(9)	790(6)	839(6)	984(6)	1888(6)	2394(6)	2397(6)	2402(6)	2589 (6)
20	387(3)	1421(3)	2066(3)	2952(3)	2955(3)	2959(3)	2990(3)	2994(3)	2999(3)

Table 3: Scores of line **5416**, which was not in our database, in [candidate (score)] format.

Rank / *Error*	1	2	3	4	5	6	7	8	9
0	828(12)	834(10)	3390(10)	4598(10)	888(9)	2317(9)	2350(9)	223(9)	4530(9)
5	801(6)	822(6)	1872(6)	1882(6)	2208(6)	2635(6)	1959(5)	2159(5)	1573(5)
10	3390(8)	2684(6)	1823(6)	4530(6)	3373(5)	62(5)	4489(5)	1636(5)	4546(5)
15	2726(5)	2838(5)	397(4)	732(4)	941(4)	1593(4)	2027(4)	2097(4)	2498(4)
20	544(9)	2027(9)	4727(8)	1482(7)	489(7)	2274(7)	2620(7)	3383(7)	3512(7)

Table 3 shows the match scores for such a text line. A combination of low scores and the similarities among the top scores give us an indication that this candidate signature is not a duplicate.

It is interesting to note that the scores for a text line that is not in the database (i.e. a non-duplicate text line) are higher than those for a text line that is in the database when 15 errors were introduced. This is due to the fact that corrupted text begins to form shape code keys that could not possibly correspond to words which appear in the English language, and thus it receives a lower score when matched against a real database. The keys in the non-duplicate text line, however, still correspond to valid keys, and receive higher scores. This gives us a quantitative measure of the similarity between signatures taken from non-duplicate, English text.

In Part II of the first experiment, we randomly chose 100 valid duplicate signatures, matched them against the 5000 signatures in the database, and recorded how often the correct match was ranked in the the top one, two, five, and ten positions. The results are shown in Table 4 for candidates corrupted with 0, 5, 10, 15 and 20 errors. We see that the correct match was consistently in the top position when there were 10 or fewer errors.

In practice, the variation between two documents which are image-variant duplicates of each other is typically due to factors such as notes, photocopying and aging, and to

Table 4: Top duplicate candidates in 100 queries

Added Errors	Top 1	Top 2	Top 5	Top 10	Top 20
0	100	100	100	100	100
5	100	100	100	100	100
10	100	100	100	100	100
15	51	58	69	77	100
20	17	21	24	30	100

Table 5: Number of duplicate candidates detected in 2500 queries from a pool of one million documents.

Added Errors	Top 1	Top 2	Top 5	Top 10	Top 20
0	2416	2443	2458	2465	2467
5	2379	2419	2447	2457	2463
8	2059	2202	2327	2390	2431
10	1403	1678	1905	2039	2181

characteristics of scanning processes including resolution, density and skew. We therefore expect few "differences" in the shape codes of duplicate documents, so the introduction of 20 errors is more then sufficient.

3.2.2 Experiment 2

Our second set of experiments involved a much larger number of signatures. We used text data similar to the data used in Experiment 1, but we used the text to create a test database of one million signatures. We then chose 5000 signatures to simulate incoming documents. 2,500 non-duplicate signatures were chosen from a different corpus and 2,500 random duplicate signatures were chosen from the Wall Street Journal corpus.

We first ran the signatures of duplicate documents through the matching process which allowed us to study the distribution of matching scores. In evaluating the results, we observed that some matches produced scores greater than the expected maximum of 50. This is because it is possible for a key to occur more then once in a signature. We also noted that some signature lines occurred more than once in the database, resulting in false duplicates. The first line of Table 5 shows the number of detected duplicates out of 2500 ranked in the top 1, 2, 5, 10 and 20 positions[1].

Next, we ran corrupted signatures of duplicate documents through the matching

[1] Some of the lines in the original database occur more than 20 times, so the "true" duplicate may not appear in the top 20. That is why even in the 0 error case, the 2500 documents were not all detected.

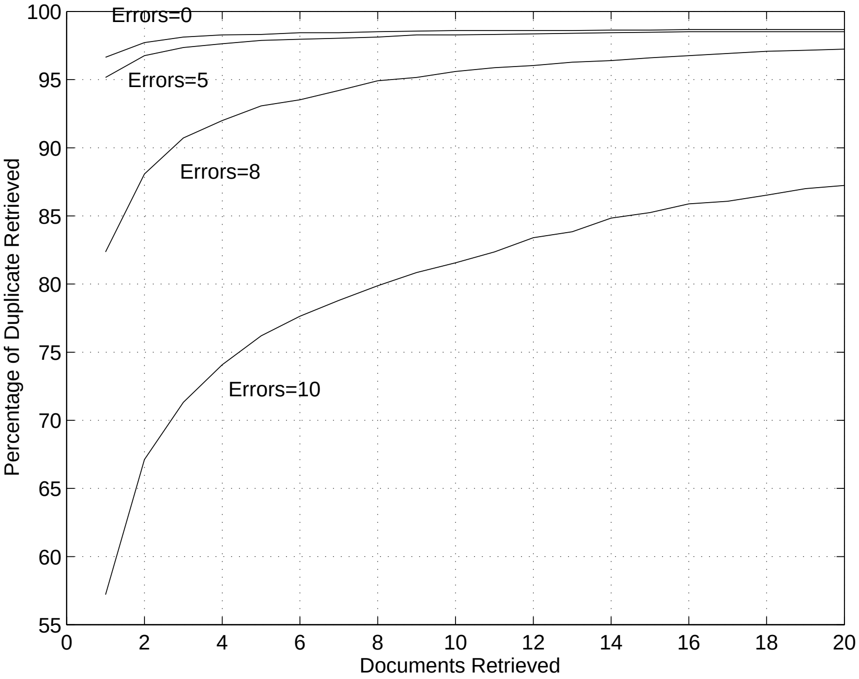


Figure 5: Percentage of duplicate documents identified in the top n candidate documents retrieved.

process, perturbing them by introducing fixed numbers of errors (5, 8 and 10). The numbers of duplicate documents ranked in the top 1, 2, 5, 10, and 20 positions for these levels of errors are shown in the bottom three lines of Table 5. Figure 5 shows the percentage of detected duplicates[2] as a function of the number of documents retrieved for each of the error levels. We see that even when there are 10 errors, in a signature of size 50 we can detect duplicates more than 84 percent of the time.

4 Implementation

Having tested the index features and the robustness of the signature matching process, the remaining task was to implement and test the line extraction and signature coding processes using image data. With degraded documents, the most critical aspect of the system is its ability to extract the same representative line from multiple instances of

[2] The "recall" of a retrieval system can be defined as the number of relevant documents which are retrieved divided by the number of relevant documents in the database. We have plotted the number of duplicates identified, divided by the total number of duplicates.

a document. Figure 6 shows a candidate document and a representative line extracted from it.

4.1 Representative Line Extraction

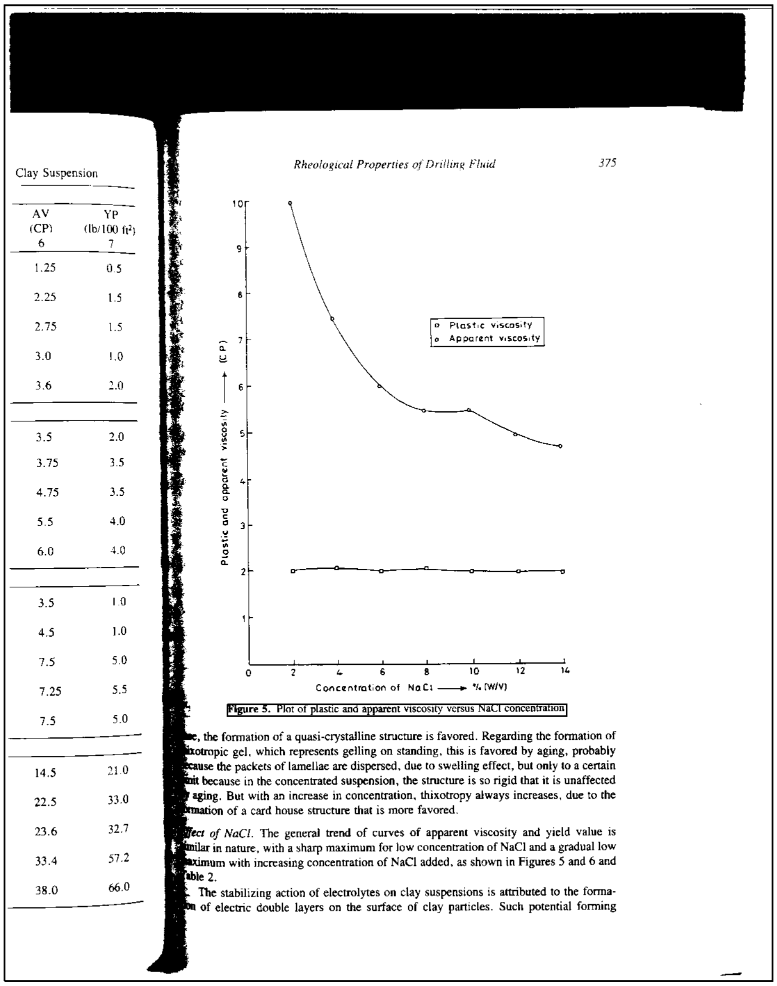
The representative line extracted from each image is a text line which has a sufficient number of characters to be used as a signature. For efficiency we divide the page image horizontally into thin "zones" and analyze the zones in vertical order from top to bottom. When a representative line is found, we do not process any portion of the image below it.

To begin, we apply a single-pass connected component algorithm to identify components within the zone of interest. To deal with noise, we want to eliminate components which appear to result from copier degradation or graphics. We use conservative thresholds on the component size to eliminate components whose sizes are less than 7pts or greater than 14pts. Although this may result in the loss of some punctuation marks and other small symbols, they are likely to be lost in both the original and duplicate documents, and not affect the match.

Next, the symbols are grouped into "words" using a smearing algorithm whose distance is a function of the average component width. Thresholds are applied to the height and width of the words to eliminate words which would not likely contribute to a unique signature.

Using a second smearing process, we group the words into a line. After line groups are formed, we begin at the top of the zone, and search for a valid line. A valid line is a line which 1) has a sufficient number of characters and 2) contains both ascenders and descenders. The first restriction insures that we have a signature which is sufficiently long, and the second restriction allows us to avoid lines such as titles which may consist entirely of capital letters, as well as lines consisting entirely of numeric data. If no valid lines are found in the current zone, the next zone is considered and the process is repeated until a valid signature line is found.

In order to avoid begin fooled by running heads or other information which is not unique to a given document, we skip the first two valid lines, and consider the third valid



Clay Suspension

AV (CP) 6	YP (lb/100 ft²) 7
1.25	0.5
2.25	1.5
2.75	1.5
3.0	1.0
3.6	2.0
3.5	2.0
3.75	3.5
4.75	3.5
5.5	4.0
6.0	4.0
3.5	1.0
4.5	1.0
7.5	5.0
7.25	5.5
7.5	5.0
14.5	21.0
22.5	33.0
23.6	32.7
33.4	57.2
38.0	66.0

Rheological Properties of Drilling Fluid 375

Plastic and apparent viscosity ⟶ (C P)
10
9
8
7
6
5
4
3
2
1
0
2
4
6
8
10
12
14
Concentration of NaCl ⟶ % (W/V)
○ Plastic viscosity
○ Apparent viscosity

Figure 5. Plot of plastic and apparent viscosity versus NaCl concentration

e, the formation of a quasi-crystalline structure is favored. Regarding the formation of ixotropic gel, which represents gelling on standing, this is favored by aging, probably ecause the packets of lamellae are dispersed, due to swelling effect, but only to a certain nit because in the concentrated suspension, the structure is so rigid that it is unaffected aging. But with an increase in concentration, thixotropy always increases, due to the rmation of a card house structure that is more favored.

ffect of NaCl. The general trend of curves of apparent viscosity and yield value is milar in nature, with a sharp maximum for low concentration of NaCl and a gradual low aximum with increasing concentration of NaCl added, as shown in Figures 5 and 6 and able 2.

The stabilizing action of electrolytes on clay suspensions is attributed to the formaon of electric double layers on the surface of clay particles. Such potential forming

Figure 6: The representative line chosen for a document with mixed text and graphics

line as the signature for the page. If this is done consistently, it will provide us with a meaningful signature to index the document.

4.2 Shape Code Extraction

Once a valid line is found, the next step is to extract its signature. The presence of factors such as uncorrected skew may cause difficulties with shape codes by extracted at the line level, so we extract the shape codes by considering a word at a time.

We begin by roughly classifying the symbols in a word into three groups by height: small symbols like punctuation, medium-height symbols such as ‘a’ or ‘c’, and tall symbols such ascenders, descenders, parentheses, etc. We use a medium-height symbol from the middle of the line to define an xheight hypothesis, using the bottom of the symbol as

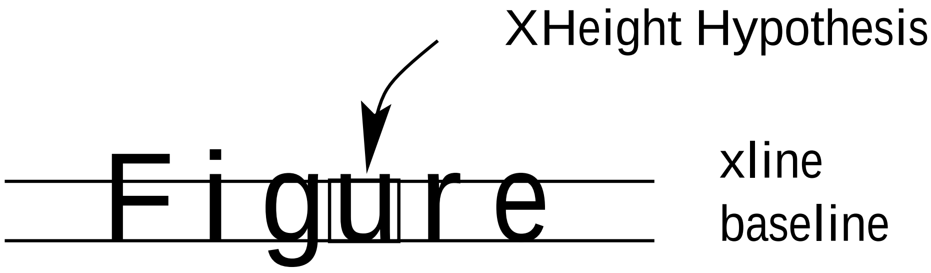


Figure 7: Example xheight hypothesis, xline and baseline

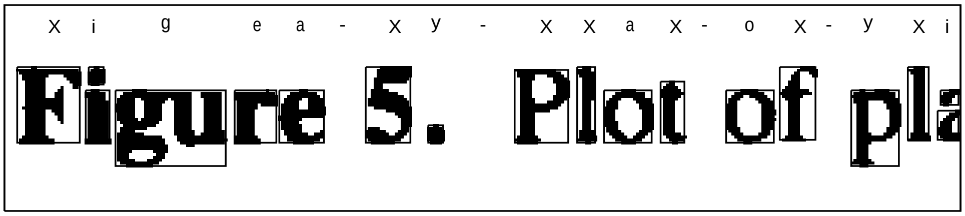
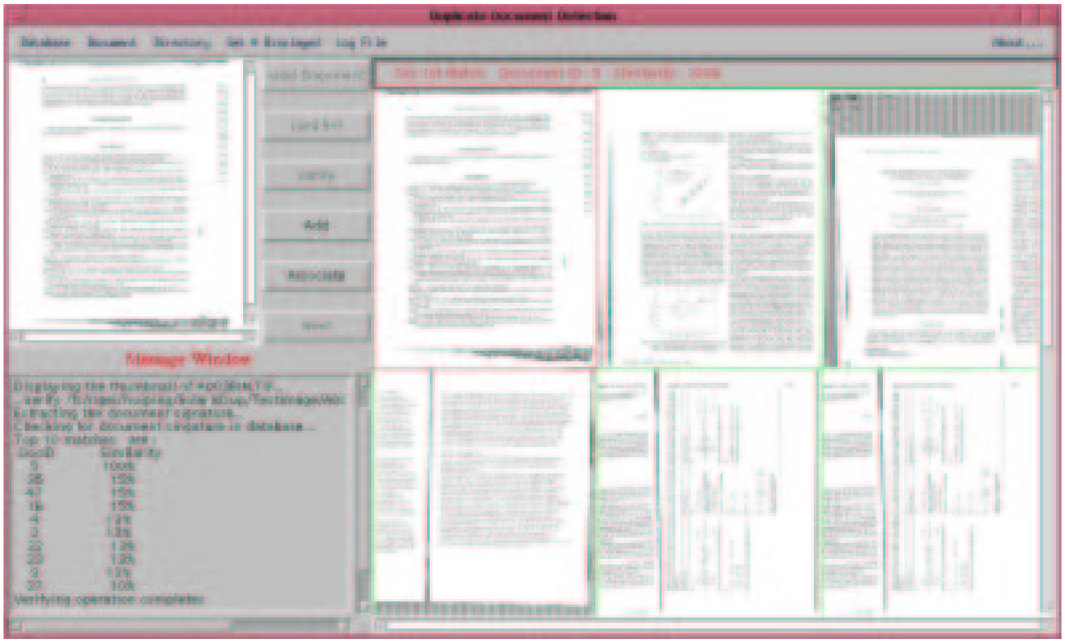


Figure 8: Shape coding results for part of the line shown in Figure 6.

the baseline and its top as the xline (see Figure 7). Starting from this seed s
encode the line outward in both directions. The idea is to adaptively use ea
and its new neighbors to predict and refine the position of the baseline on eacl
If the symbol is a punctuation mark (i.e. it does not span the region between t
and the xline, or it extends into both the ascender and descender regions),
but do not adjust the x or baselines. If the character is an ascender, we so c
adjust the baseline to the bottom of the character. If it is a descender, we so
adjust the xline to its top. Finally, if it covers only the xheight region, we a
the xline and the baseline.

After all the characters in the word are classified, holes are identified and t
are refined if necessary. Figure 8 shows an example of the results of coding.

The shape coding process is very accurate for clean data which is not sig
skewedA problem arises when many characters touch; the system may skip an ot
valid line. This tends to occur when documents are photocopied repeatedly,
thresholded, or are significantly degraded physically. For example, if the o
printed with a laser printer, and our duplicate candidate is a third-generation
enough symbols may touch to put us below the signature length limit. But as
in Section 3.2, even when there are 10-15 shape code errors, we can still detect



```
Displaying the thumbnail of [illegible]
...verify [illegible]
Extracting the document signature...
Checking for document signature in database...
Top 10 matches: [illegible]
 DocID      Similarity
   5          100%
  35           15%
  47           15%
  16           15%
   4           12%
   3           13%
  22           12%
  23           12%
   3           12%
  20           10%
Verifying operation complete.
```

Figure 9: User interface.

4.3 Interface

A user interface has been implemented in Java to provide users with an easy way to interact with a database (Figure 9). It allows the user to select a database and verify either a single document or group of documents. The retrieved documents are ranked and presented for confirmation along with a quantitative measure of similarity. If the candidate document is not found in the retrieved set, the user can add it to the database. Otherwise, the user can select a retrieved document and mark it as a duplicate, mark it as "similar" or mark it as unknown. Log files are generated to track documents through processing and indexing.

The computational requirements for the system are reasonable. The image processing, including thumbnail generation, takes about 7 seconds per document, indexing into a database of one-million documents takes another 2 seconds, and displaying 20 candidates takes another 2 seconds with unoptimized code and a SPARC Ultra-1. The first two phases can be performed offline in batch mode.

Table 6: Results of matching 307 duplicate document images against a database of approximately 1000 documents.

	Top 1	Top 2	Top 5	Top 10	Top 20
Number of Correct Identifications	286	296	298	302	307
% of Total	93.2	96.4	97.1	98.4	100.0

5 Experiments with real images

For our experiments with real document images, we used the University of Washington Document Image Databases [4] in which the data consist primarily of technical journal image pages scanned from first and third generation photocopies of the original documents, and memos scanned directly from original documents.

In an initial experiment, 1035 image documents were inserted into a database. As each new document was added, the top-ranked 20 documents were returned along with their match scores. As expected, the average match score was low (20%). The average difference between the match scores of the top two matched database documents was also low (about 4%); thus no retrieved document stood out as significantly more similar to the candidate document.

We then tested 307 duplicate documents which were third-generation photocopies of the originals[3]. Some of these copies were significantly degraded. For these duplicate documents the verification procedure was followed, but without adding the candidate documents to the database. The top 20 database matches were determined for each candidate document, along with their match scores. The average score for the top match was over 75% and the average difference between the scores of the top two matches was over 48%. The average match score for the second-ranked match was only about 20% and similar to the scores of the top-ranked non-duplicates. Clearly, the duplicates tended to have significantly higher match scores than the second-ranked non-duplicate documents. Table 6 shows the distribution of rankings for the 307 duplicates.

All of the errors in the top-ranked documents were due to significant numbers of merged characters which resulted in missing the representative line completely. We are

[3] These photocopies were also present on the UWASH CDROM.

testing an improved character segmentation scheme which is not based entirely on white space, but also uses character width statistics. We anticipate that this approach will reduce the errors by as much as 75%.

It should be pointed out that our experiments have used only the shape code features for document identification, and have not considered other features such as the number of pages in the document, the number of lines on a page, or the density of the page, for example. A more advanced system could also make use of such features.

6 Discussion and Conclusions

The problem of duplicate document image detection is of great importance, not only within a single database, but also for image query engines of the future which operate across multiple databases.

If we can reduce the number of duplicate document candidates to a manageable size, more refined algorithms that directly compare the images can be used, or thumbnails of the top 20 candidates can be rapidly presented to an operator to verify that a duplicate exists. One advantage of not using structural information for indexing is that non-duplicate candidates tend to be visually very different from the query document and can be eliminated rapidly by the operator.

The novel approach to the problem of duplicate document detection described in this paper shows great promise, as demonstrated both by the simulation results on a million documents (Section 3.2) and the experimental results on a small image database (Section 4). We can deal more robustly with small image distortions and have the advantage over competing approaches that we do not need any content-level information, either a priori or as part of the analysis process.

It appears that a system could be developed within current limitations on system resources which would provide a cost-effective solution to the problem. It is estimated that in some large applications, as much as 25% of the cost could be saved by identifying duplicate documents.

References

[1] D. Doermann, H. Li, and O. Kia. Duplicate document image detection. Technical Report CS-3739, University of Maryland, College Park, 1997.

[2] J.J. Hull. Document image matching and retrieval with multiple distortion-invariant descriptors. In *Proceedings of the International Workshop on Document Analysis Systems*, pages 383 – 400, 1994.

[3] N. Shivakumar and H. Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries*, 1995.

[4] University of Washington. Document image database collection. CDROM.

[5] A. Parker and J.O Hamblen. Computer algorithms for plagiarism detection. *IEEE Transactions on Education*, pages 94–99, 1989.

[6] H. Garcia-Molina S. Brin, J. Davis. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Annual Conference*, 1995.

[7] A.L. Spitz. Using character shape codes for word spotting in document images. In *Shape, Structure and Pattern Recognition*, pages 382–389. World Scientific, Singapore, 1995.

[8] H. Tanaka and A. Kogawara. High speed string edit methods using hierarchical files and hashing technique. In *Proceedings of the International Conference on Pattern Recognition*, pages 334–336, 1988.

[9] T. Yan and H. Garcia-Molina. Duplicate detection in information dissemination. In *Proceedings of the Very Large Database Conference*, 1995.

[10] U.Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Conference*, pages 1–10, 1994.

A Feasibility Analysis

This appendix presents a theoretical feasibility analysis demonstrating that the system design is realistic and that it is robust to anticipated errors in the signature extraction. The analysis assumes we have extracted a candidate signature. The signature is a vector of feature values used to represent the document, and a key, possibly resulting from a partitioning of the signature, is used to index into the database. The parameters listed below will be used in the feasibility and performance analysis of the algorithm. A majority of the parameters reflect physical constraints of the system and are necessary to explore scalability.

System-Dependent Parameters

F : Maximum allowable number of files in the operating system.

S : Maximum allowable size for a single file in the operating system.

K : Main memory size used for processing buckets.

t_d : Disk access time in sec.

t_m : Memory access time in sec.

t_t : Disk transfer rate in bytes/sec.

Algorithm-Dependent Parameters

N : Number of documents.

N_f : Maximum number of files for the index table.

S_f : Maximum file size for storing b buckets.

E : Size of index table entries in bytes.

d : Average size of documents in bytes.

l : Number of buckets that will be kept in the main memory.

b : Number of index table buckets that will be stored in a file in main memory.

Independent Variables

a : Size of alphabet used to construct the signature.

m : Size of the signature.

w : Window size.

k : Number of keys for a given signature size $(m - w + 1)$.

A.1 Analysis of Indexing

The matching algorithm relies on an index structure that is generated as documents are added to the database. Each signature is partitioned into equal-sized overlapping windows of size w, to be used as index keys. This partitioning provides robustness in the sense that errors in the signature will only be propagated within the window, but the smaller key size results in a less unique set. For a signature of size m, $k = (m - w + 1)$ possible keys (Figure 2 in the main text) must be indexed. The index table has on the average $k \times (\frac{N}{a^w})$ entries per bucket with each entry being the identification of a document which contains that key. Figure 10 shows the index table structure.

An array with a maximum size of min $\{N, (k^2 \times (\frac{N}{a^w}))\}$ keeps a count of the documents which contain keys that have matched to the input key. When the document's keys are indexed, a counter is incremented for each document which contains that key. The most frequently occuring documents are then identified as candidate duplicates. A further level of refinement can be achieved by using a more elaborate matching algorithm that includes the positional matches between the keys, but this is not covered in this analysis.

The search time for the matching operation can be generalized as follows[†] :

$$\begin{aligned} \text{Worst case search time} &= \text{Bucketsearch} + \text{Diskoperations} \quad (1) \\ &= k \times \left(\left(k \times \frac{N}{a^w} \times t_m \right) + \left(t_d + \frac{b \times E \times \frac{k \times N}{a^W}}{t_t} \right) \right) \end{aligned}$$

[†]: If the bucket size stored in the file is greater than K, then to search the entire file requires multiple

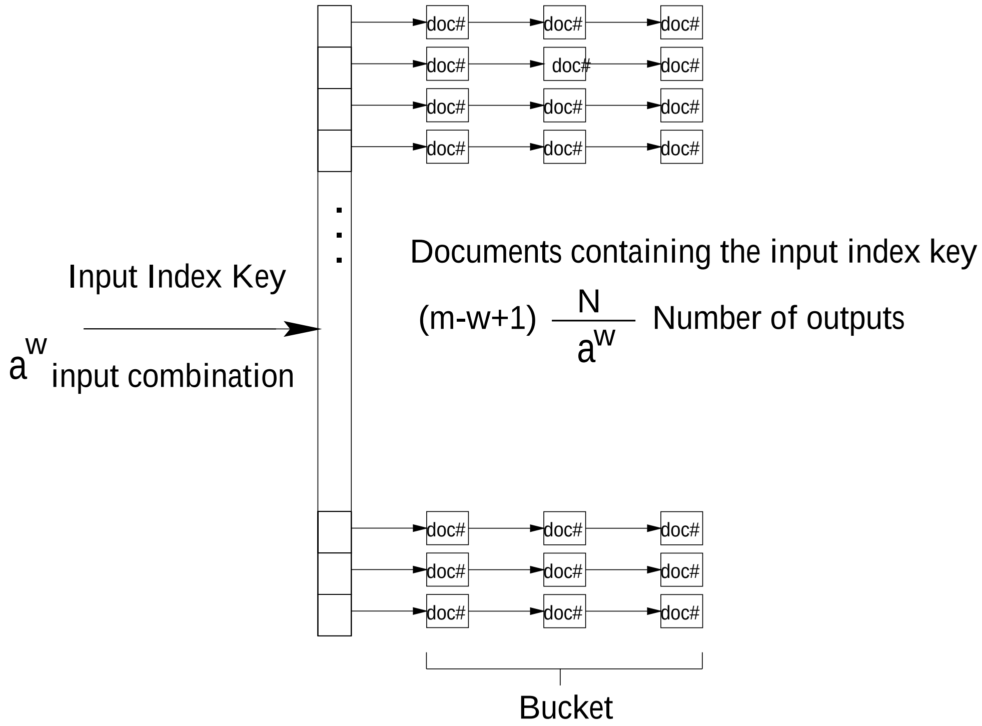

doc#
Documents containing the input index key
(m-w+1) N/a^w Number of outputs
Input Index Key
a^w input combination
Bucket

Figure 10: The index table.

Note that every document signature has k index terms, and each bucket has an average of $(k \times \frac{N}{a^w} \times t_m)$ entries. Here, the disk access time (t_d) is multiplied by the number of windows, since in the worst case, every indexed bucket is in a separate file. The transfer time (t_t) is the time required for reading the buckets from the disk, and since disk operations are more expensive than memory operations, the number of buckets that are stored in a single file should be optimized to decrease disk access time. We can choose b such that the file size for the b buckets is not greater than the main memory size $(l \leq b)$, thus minimizing the disk access time.

These parameters should also be chosen not to exceed the limitations of the operating system. There is an upper limit on the maximum allowable number of files (F) and on the maximum allowable file size (S) in every operating system. Since we have a large amount of data, the index table structure should be within these limits even if we have

disk accesses. Here the disk access time (t_d) is assumed to be 9 ms. The memory access time (t_m) is 100 ns and the transfer rate (t_t) is assumed to be 15 MB per sec.

to partition the index table. The maximum number of files to store index buckets is

$$N_f \;=\; \frac{a^w}{b} \leq F \tag{2}$$

(the total number of buckets (a^w) divided by the number of buckets in a single file (b)) where F is the operating system restriction on the maximum number of files. The maximum file size is then

$$S_f \;=\; \left(\frac{k \times N}{a^w} \times b \times E\right) \leq S \tag{3}$$

or the number of entries per bucket multiplied by number of buckets in each file and the entry size in bytes. Since each bucket has a maximum of $\frac{(k \times N)}{a^w}$ entries, if we keep l buckets in memory, then the following constraint should be satisfied:

$$(l \times \frac{k \times N}{a^w} \times E) \leq K. \tag{4}$$

The construction of the index table should satisfy (2), (3) and (4). The overhead associated with the index table can be estimated as

$$\text{Overhead} \;=\; \frac{k \times E}{d} \tag{5}$$

Note that equation (5) is independent of N, the number of documents in the database.

A.2 Performance Analysis

To characterize the performance of a given system we must first formulate several probabilistic models and define a design criterion. This is done by using hypothesis testing and deriving an appropriate distribution function such that specific performance measures can be computed. For a hypothesis test consider two events, the null event and the alternative event. The Null event, H_0, is that there is no duplicate for a given document, and the Alternative event, H_1, is that there is a duplicate.

$$\begin{aligned} H_0 : &\quad \text{Null Hypothesis} \quad & P(\eta|H_0) \\ H_1 : &\quad \text{Alternative Hypothesis} \quad & P(\eta|H_1) \end{aligned} \tag{6}$$

Associated with each hypothesis is a probability measure η, the probability of detection. For detection, a threshold is used to determine which hypothesis is valid. In this case

η represents a matching score assigned to an observation. By choosing an appropriate threshold η_T we make the decision that if $\eta \geq \eta_T$ we accept the Alternative hypothesis, and if $\eta < \eta_T$ we accept the Null hypothesis. The probability of detection and the probability of false alarm capture the performance of this detector:

$$\begin{aligned} P_D(\eta_T) &= P(\eta \geq \eta_T | H_1) \\ P_{FA}(\eta_T) &= P(\eta \geq \eta_T | H_0) \end{aligned} \tag{7}$$

These numbers will be used as operating specifications given various parameters, but to fully specify P_D and P_{FA} we need to analyze or make assumptions about the data and the matching processes.

Given a signature of size m from an alphabet of size a and an observation (or index key) of size w, define x_i to be the ith symbol and y_j to be the jth index key. With $x_i \in X_i = \{0,1\}^{\log_2(a)}$ it is easy to see that all the x_i's are independent. In fact

$$P(X = x) = \prod_{i=1}^{m} P(X_i = x_i) \qquad \text{for } x \in X = \bigcup_{i=1}^{m} X_i = \{0,1\}^{m \log_2(a)} \tag{8}$$

Let us define $y_i = \{x_i, x_{i+1}, ..., x_{i+w-1}\}$; y_i is clearly dependent on y_{i+1} to y_{i+w-1}. As described in the previous sections, the y_i's will be used for indexing into a signature database and the number of hits determines η. When indexing, however, there is no dependence on the order of the observed y_i's, and in a hypothetical situation two documents could have large numbers of hits resulting from matching y_i's out of order. Smaller keys would increase this effect and larger keys would decrease it; In most cases, however, we can argue that the y_i values used to calculate η are independent. We shall assume that it is equally likely (with probability $\frac{1}{a^w}$) to observe any y_i.

Given these conditions, a signature contains a set of features y_i for $i = 1, 2, ..., k$. Since each y_i is now treated as an independent observation and we have k observations, we can compute the probability of η matches out of k trials to be binomial, $b(k, \frac{1}{a^w})$:

$$P(\eta) = \begin{pmatrix} k \\ \eta \end{pmatrix} \left(\frac{1}{a^w}\right)^{\eta} \left(1 - \frac{1}{a^w}\right)^{k-\eta} \tag{9}$$

Equation (9) is the probability of the number of matches given the Null hypothesis, $P(\eta|H_0)$. This probability measures the relationship between different signatures and is relatively easy to understand and compute. The assumption that the occurrences of keys are equally likely, however, is unrealistic. For example, it is unrealistic to have all descenders in a window observation. We therefore attempt to obtain realistic probabilities of key occurrences empirically and revise the probability distribution function of the NULL hypothesis. The probability measure for the Alternative hypothesis, however, is not so trivial.

To derive the probability distribution of the Alternative hypothesis, we must consider what realistic errors may be observed between a candidate signature and its corresponding entry in the database. The discrepancies between observed and recorded signatures can be characterized as insertions, deletions, or substitutions of shape codes. In general, errors increase with degradation, so we assume that the probability of observing i errors is a decaying exponential function, defined by a decay rate β . The general form of this function is $P(i) = P(0)e^{-\beta i}$, where $P(0)$ is calculated given $\sum_{i=0}^{m} P(i) = 1$. Calculating $P(0)$ and formulating the distribution function we get

$$P(i) = P(0)e^{\beta i} = \left(\frac{1 - e^{-\beta}}{1 - e^{-\beta(m+1)}}\right) e^{-\beta i} \quad \text{for } i = 0, 1, ..., m. \tag{10}$$

This equation is approximate since this probability measure depends on the accuracy of identifying shape codes, the statistics of shape codes, and the similarity of duplicates found in the database. It is, however, impossible to characterize the statistical nature of these measures, so we have simplified the expression to an exponential function. In order to formulate the probability distribution of the Alternate hypothesis we need to study the relationship between the number of errors and the matching score. In the case where a single error occurs, the error is propagated to w of the k index entries. This results in a matching score of $k - w$ out of a maximum of k. Although the errors might occur at either end of the signature, and insertions and deletions may change the matching score, these occurrences are statistically insignificant and offset each other. For multiple errors, the worst-case scenario is when errors occur w shapes away from each other so the errors propagate to the most keys. For this worst-case scenario, i errors will translate to a maximum matching score of

$$\eta_{worst} \;=\; \begin{cases} k - iw & \text{for } i = 0, 1, ..., \lfloor \frac{k}{w} \rfloor \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

For the best-case scenario, the errors could occur next to each other, yielding a maximum matching score of

$$\eta_{best} \;=\; k - \left\lceil \frac{i}{w} \right\rceil w \approx k - i \quad \text{for } i = 0, 1, ..., k. \tag{12}$$

The probability that worst-case or best-case errors occur is clearly dependent on the number of errors. This is true since, by definition, $i = 1$ results in a lower bound and $i = m$ results in an upper bound on the matching score. It is also true that the score probabilities change as a power function of the number of errors, and their analysis is extremely complex. Therefore, for simplicity, we will assume that the matching score is an average of the worst and best case scenarios:

$$\eta(i) \;=\; \frac{\eta_{worst} + \eta_{best}}{2} \;=\; \begin{cases} k - \frac{i + i \times w}{2} & \text{for } i = 0, 1, ..., \left\lfloor \frac{2k}{w+1} \right\rfloor \\ \frac{k-i}{2} & \text{for } i = \left\lfloor \frac{2k}{w+1} \right\rfloor + 1, ..., m - w + i \end{cases} \tag{13}$$

Using Equations (10) and (13), the probability distribution function of the Alternative hypothesis is fully specified. We are now able to calculate the probabilities of detection and false alarm.

Given a threshold η_T, any document with a higher score is identified as a duplicate, as shown in equation (7). We must now apply this criterion to the probability distribution functions of the Null and Alternative hypothesis to get the probability of false alarm and probability of detection, respectively. Equation (9) shows the distribution function of the Null hypothesis, and summation over $\eta = \eta_T, ..., k$ gives the probability of false alarm:

$$P_{FA} = P(\eta \geq \eta_T | H_0) = \sum_{\eta=\eta_T}^{k} \left(\begin{array}{c} k \\ \eta \end{array} \right) \left(\frac{1}{a^w} \right)^{\eta} \left(1 - \frac{1}{a^w} \right)^{k-\eta} \tag{14}$$

In order to calculate the probability of detection we need to consider the inverse of equation (13) to be used in equation (10). Using η_T and solving for i, the maximum number of allowable errors such that the effective score equals the threshold score:

$$i_{max} \;=\; \begin{cases} \frac{2(k-\eta_T)}{1+w} & \text{for } \eta_T \;>\; \frac{k(w-1)}{2w} \\ k - \eta_T & \text{otherwise} \end{cases} \tag{15}$$

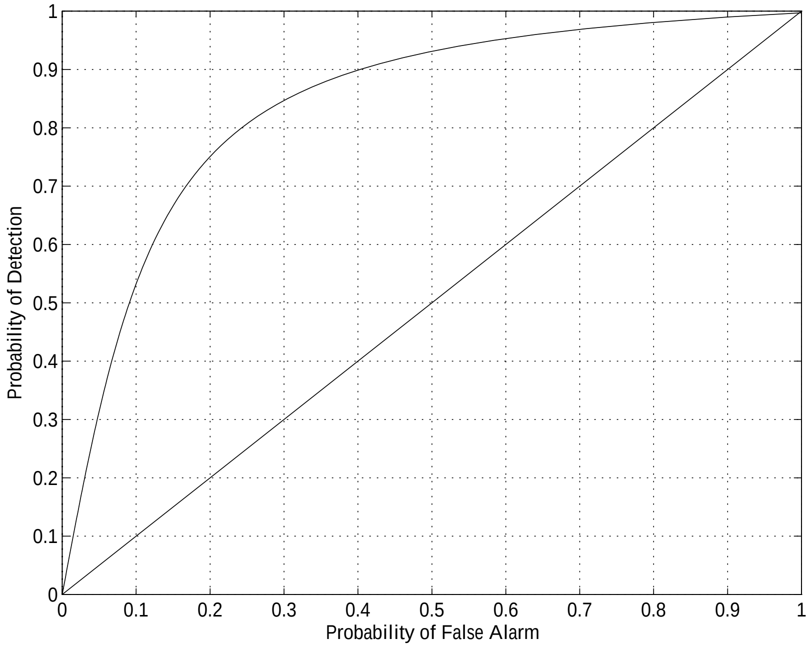


Figure 11: Typical ROC Curve

Then summing equation (10) from zero to i_{max} gives the probability of detection:

$$P_D = P(\eta \geq \eta_T | H_1) = \sum_{i=0}^{i_{max}} \left(\frac{1 - e^{-\beta}}{1 - e^{-\beta(m+1)}} \right) e^{-\beta i} \tag{16}$$

Given equations (14) and (16), it is possible to plot the probability of false alarm with respect to the probability of detection as a function of the threshold η_T. This plot is traditionally called the ROC (Receiver Operating Characteristic) curve; it is shown in Figure 11 for a typical case. In the pioneering days of radar technology, microwave engineers drew such curves for a number of signal power levels to determine the optimum operating point. We will perform a similar task, but instead of varying signal strength we will vary signature length, alphabet size, and window size. Figure 11 shows a diagonal line which signifies the minimum achievable performance. The diagonal line represents picking the Null or the Alternative hypothesis based on a coin toss; if an algorithm performs below this line, a coin toss would perform better.

A.3 Operational Scenarios

The following example represents a possible document management scenario. If $N =$ 50 million, each document has an average of 10 pages and each page (TIFF file) has an average size of $100KB$, so that the size of the database is approximately $50,000$ GB, assuming it is necessary to keep all images. The index table has size $46 \times 50\text{M} \times 4\text{Byte} =$ 8.6GB assuming $N = 50\text{M}$, $m = 50$, $w = 5$, and $a = 8$ since each document will be indexed by $k = 46$ keys. This means it is not possible to store the table in main memory. A 4 Byte representation of the document identification number can be used for the 50M documents. Each of the 32K buckets in the index table has an average of 69K document identification entries.

This scheme has two limitations that are imposed by the UNIX file system. The first is that the size of the index table cannot exceed the size of main memory, so disk caching is required. The second limitation is that the entire table cannot fit within a single file (8.6 GB), so it must be divided into a number of smaller files that are within the limits of the maximum file size imposed by UNIX.

Table 7 shows the index table size and the number of entries per bucket for different numbers of documents. To calculate the search time we use $t_d = 9$ msec, $t_m =$ 100 nanosec, and $t_t = 15$ MB/sec.

N (million)	Entries per bucket	Index table size in GB	Expected search time in sec
10.0	14039	1.7	0.64
20.0	28077	3.4	0.87
30.0	42115	5.1	1.10
40.0	56153	6.9	1.33
50.0	70191	8.6	1.56

Table 7: For $m = 50$, $w = 5$, $a = 8$ the index table has 32K buckets.

Table 8 shows index table characteristics and an estimate of the running time of the algorithm for different alphabet sizes. In the first two rows, the search time is larger than the other values, due to the large bucket size. This requires multiple disk accesses to search all of the buckets. However, in the other cases the whole bucket can be stored in main memory, requiring only one disk access. We can see from Table 8 that a smaller bucket size is desired for faster search. But this can only be achieved by increasing a, which means greater preprocessing time for extracting shape codes, and possibly decreases the robustness.

a	Number of buckets	Entries per bucket	Bucket size (MB)	Buckets per file	Bucket file size (MB)	Number of files	Expected search time (sec)
4	1024	2246094	8.6	1	8.6	1024	38.68
5	3125	736000	2.8	1	2.8	3125	12.82
6	7776	295782	1.1	1	1.1	7776	5.23
7	16807	136848	0.5	3	1.6	5603	2.64
8	32768	70191	0.3	7	1.9	4682	1.56

Table 8: Index table characteristics for $w = 5$, $m = 50$, $N = 50\text{M}$, $K = 2\text{MB}$

Table 9 shows index table characteristics and an estimate of the running time of the algorithm for different values of w. The number of documents is 50M, $m = 50$, $a = 8$. The first two cases have significant search time ranges due to the large bucket size. We can see from Table 9 that the search time decreases as w gets larger, but a larger w means a more error-prone system.

w	Number of buckets	Entries per bucket	Bucket size (MB)	Buckets per file	Bucket file size (MB)	Number of files	Expected search time (sec)
3	512	4492188	17.136	1	17.1	512	76.94
4	4096	561524	2.142	1	2.1	4096	9.98
5	32768	70191	0.268	7	1.9	4682	1.56
6	262144	8774	0.033	59	2.0	4444	0.56
7	2097152	1097	0.004	477	2.0	4397	0.43
8	16777216	138	0.001	3799	2.0	4417	0.42

Table 9: Index table characteristics for $a = 8$, $m = 50$, $N = 50\text{M}$, $K = 2\text{MB}$

Table 10 shows index table characteristics and a rough estimate of the running time of the algorithm for different values of m. The number of documents is 50M, $w = 5$, $a = 8$. As m gets larger, the hash table size increases, which increases search time.

By looking at Tables 8-10 and the ROC curves in Figures 12-14 we can find "good"

m	Number of buckets	Entries per bucket	Bucket size (MB)	Buckets per file	Bucket file size (MB)	Number of files	Expected earch time (sec)
50	32768	70191	0.268	7	1.9	4682	1.56
60	32768	85450	0.326	6	2.0	5462	2.20
70	32768	100709	0.384	5	1.9	6554	2.95
80	32768	115967	0.442	4	1.8	8192	3.81
90	32768	131226	0.501	3	1.5	10923	4.77
100	32768	146485	0.559	3	1.7	10923	5.85

Table 10: Index table characteristics for $a = 8$, $w = 5$, $N = 50\text{M}$, $K = 2\text{MB}$

values for the model parameters for the case of 50 million documents. From Table 8 and Figure 12 we can see that although an increasing a has a great effect on the search time, its effect on the detection probability is not significant. In that sense, choosing a proper value for a will determine search time without effecting performance. The small values of m are better for both search time and detection probability. Choosing a best value for w requires more thought, since detection probability is inversely proportional to both search time and w. Picking a value for w is thus a design issue: giving preference to search time or to probability of detection.

To summarize, the effect of a, the size of the alphabet, can be seen from Figure 12. The increasing value of a does not change the probability of detection significantly. But from Table 8 we can see that the effect of a on the search time is great.

Figure 13 shows the effect of changing w, the window size. Increasing w makes the detection probability worse. However a very small value (3,4) for w increases the search time, since the bucket size increases significantly, requiring multiple disk accesses.

Figure 14 shows the effect of changing m, the signature size. Increasing m makes the detection probability higher.

The surfaces defined by search time, detection probability and false alarm probability can help the designer visualize the "relative goodness" of the operating point (Figure 15). For detection we can see that lower values of w and higher values of m are desired. However, we can see that the probability of false alarm increases for low values of w and high values of m. The search time favors high values of w and low values of m.

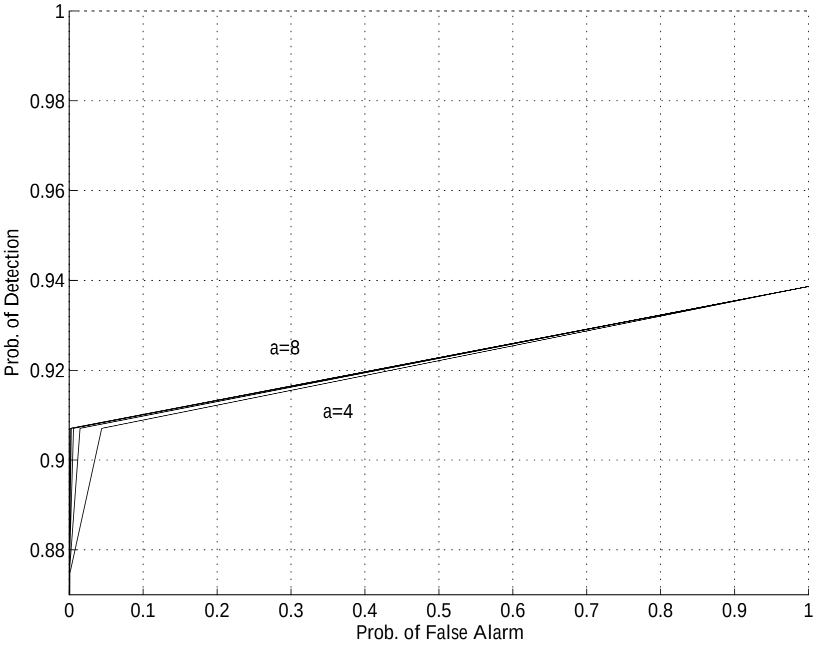


Figure 12: ROC Curves for $\beta = 0.01$, $m = 50$, $w = 5$ and various values of a (4,...,8).

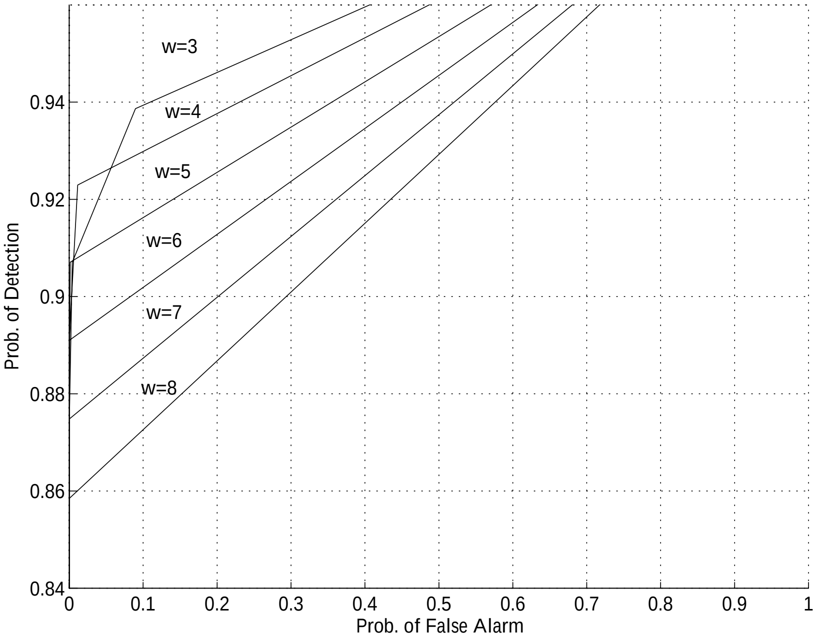


Figure 13: ROC Curves for $\beta = 0.01$ $m = 50$, $a = 8$ and various values of w (3,...,8).

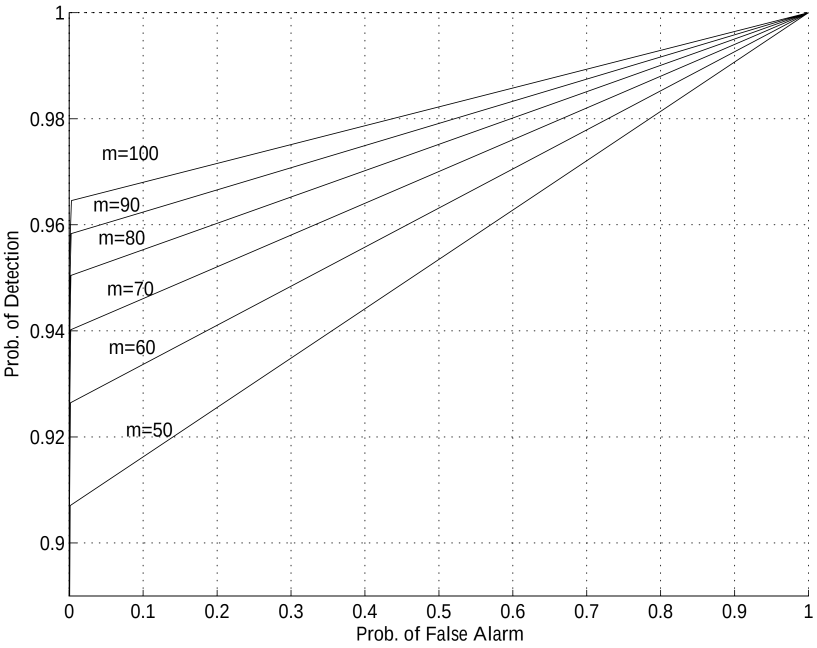


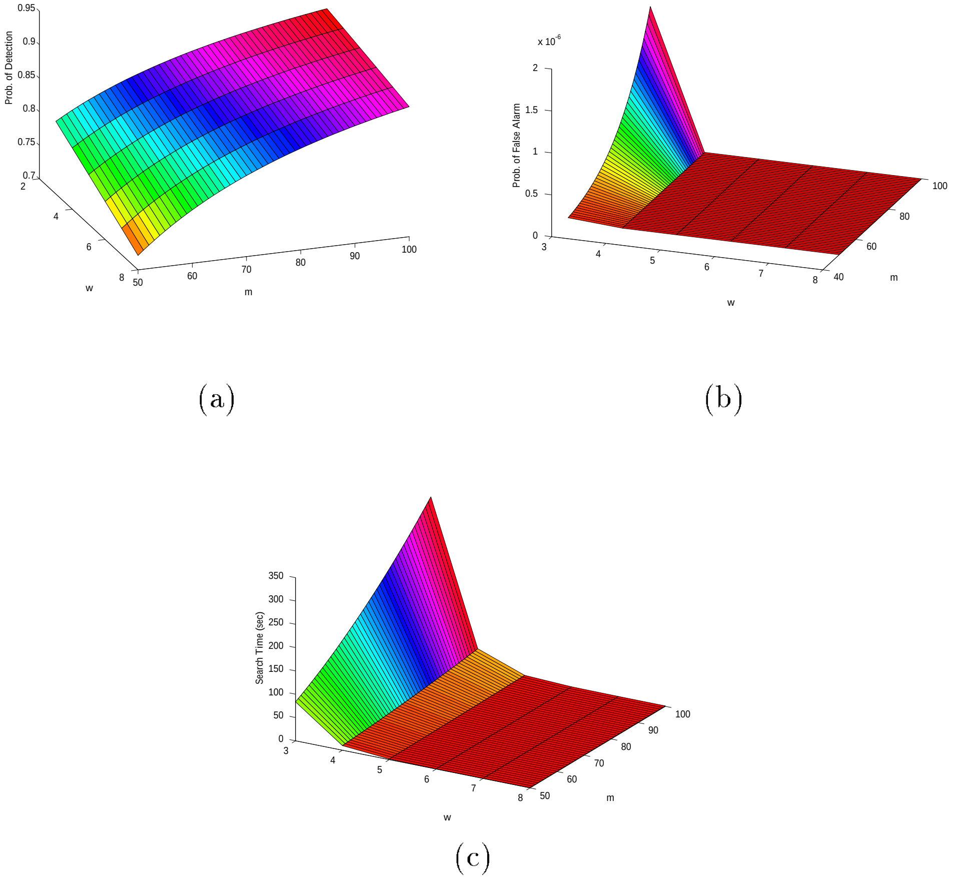
Figure 14: ROC Curves for $\beta = 0.01$ $w = 5$, $a = 8$ and various values of m (50,...,100).

A.4 System Design

Our model has two main profiles: performance and system resources. The performance can be characterized by specifying the error tolerance. The system resources can be thought of as defining constraints on the desired performance. It is hard to find a formula that can give us optimum values of the model parameters (m, w, a) for given system resources and performance range. But we can at least give a recipe for finding "good" values for the parameters. Once the system resources are characterized, making tables (like Tables 8 and 9) for different combinations of the parameters (m, w, a), and considering constraint equations (2-4) for different values of these parameters, helps us understand the possible ranges of the parameters. Then, by drawing ROC curves and looking for a desired detection probability range, we can shrink the ranges of the parameters further.

Let us comment on each parameter's effect. The increase in the value of m (the size of the signature) increases the search time and the size of the index table. However, the probability of duplicate detection becomes higher (Figure 14).

The increase in the value of w (the window size) makes the detection probability



(a)

(b)

(c)

Figure 15: (a) Probability of detection, (b) probability of false alarm, and (c) search time as functions of m and w.

lower (Figure 13). We can conclude from Figures 13 and 14 that only $k = m - w + 1$ (the number of keys) matters for the detection probability, because increasing w means decreasing k, which lowers the detection probability.

The value of a (the alphabet size) has no significant effect on the detection probability (Figure 12). But increasing it improves the detection probability, since the false alarm rate drops.

The choice of a and w is crucial for the number of entries per bucket. The number of entries per bucket affects the search time, and as the search time increases, the number of entries increases. As w or a increases, the number of entries per bucket decreases, but the errors increase. Smaller values of w are desirable for robustness.